Naval Research Laboratory
Washington, DC 20375-5320

NRL/FR-MM/6180--00-9939

A Multimodal Virtual Environment for Ship Familiarization

DAVID L. TATE

Navy Technology Center for Safety and Survivability
Chemistry Division

STEPHANIE S. EVERETT
TUCKER MANEY
KENNETH WAUCHOPE

Navy Center for Applied Research in Artificial Intelligence
Information Technology Division

May 12, 2000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 12, 2000	Final Report 3/4/99-9/30/99

4. TITLE AND SUBTITLE

A Multimodal Virtual Environment for Ship Familiarization

5. FUNDING NUMBERS

6. AUTHOR(S)

David L. Tate, Stephanie S. Everett,* Tucker Maney,* and Kenneth Wauchope*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR-MM/6180--00-9939

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*Navy Center for Applied Research in Artificial Intelligence
Information Technology Division

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

When officers and crew first report for duty on a ship, they are unfamiliar with the ship's layout and the locations of the various offices and spaces they will encounter in their normal, day-to-day activities. Through an inefficient, time-consuming process that may span several months, they learn the locations of areas important to them by asking directions, following others, or sometimes just trial and error. This report describes the initial prototype development of a new method of familiarizing one's self with a ship by using a multimodal, immersive, virtual environment. This tool combines virtual reality, speech recognition, natural language processing, route planning, and speech synthesis to create an automated stand-alone, self-paced system that sailors can use as a wayfinding aid to help them rapidly develop good familiarity with the ship's spaces.

14. SUBJECT TERMS

Virtual environments
Virtual reality
Mutimodal interfaces
Speech interfaces
Speech recognition
Natural language processing
Route planning
Wayfinding

15. NUMBER OF PAGES

16

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

A MULTIMODAL VIRTUAL ENVIRONMENT FOR SHIP FAMILIARIZATION

INTRODUCTION

When officers and crew first report for duty on a ship, they are unfamiliar with the ship's layout and the locations of the various offices and spaces they will encounter in their normal, day-to-day activities. Through an inefficient, time-consuming process that may span several months, they learn the locations of areas important to them by asking directions, following others, or sometimes just trial and error. Even after viewing the ship's diagrams, a conceptual model of the ship is usually not acquired until the space can be explored. A new method of acquiring this conceptual model is needed to eliminate the delays encountered by new crew members.

Unfamiliarity is a problem encountered by all sailors as they rotate through their individual duty assignments, but new ships are particularly subject to this problem because the entire precommissioning crew is unfamiliar with the ship. If a method were available that allowed the crew to familiarize themselves with the ship before reporting aboard, much of the disorientation and confusion of being in unfamiliar surroundings would be eliminated.

BACKGROUND

The advancement of high-performance computer graphics systems has spawned a relatively new technology referred to as virtual reality (VR) or virtual environments in which computer-generated "worlds" can be explored interactively. These virtual environments integrate high-fidelity computer graphics with intuitive user interfaces to provide a useful system for realistic simulation of environments that are either models of actual environments, or complete synthetic creations. Depending on the degree of realism that is needed for any particular simulation, the amount of immersion the virtual environment provides can range from a simple desktop display to multiuser, fully immersive, room-sized displays. One of the most common methods for providing a fully immersive environment while maintaining a reasonable budget is to use a head-mounted display (HMD) that renders the user's view based on the tracked position and orientation of the HMD. Regardless of the type of display or the amount of graphics processing power used, VR is becoming a useful tool for developing a variety of synthetic virtual environments.

The entertainment industry has developed many successful VR games that provide intense action in fictitious fantasy worlds, but the Navy is interested in virtual environments that can be used to effectively train sailors to handle unexpected problems that may arise while aboard ships [1]. VR provides a good way to study three-dimensional (3-D) environments, and it is often used in architectural walk-throughs and for virtual prototyping to allow designers, architects, and endusers to get a good idea of what the real environment will look like based on their exploration of the virtual environment. Using virtual environments for familiarizing people with real physical spaces has been validated through experimental studies showing that immersive virtual environments are effective tools for acquiring conceptual models of 3-D physical spaces [2, 3, 4].

Manuscript approved March 2, 2000.

Our previous work showed that a measurable performance improvement could be achieved by shipboard firefighters that used immersive virtual environments as part of their mission preparation [5, 6]. Feasibility tests were conducted aboard the ex-USS *Shadwell* [7] to determine if virtual environment could be used to improve the performance of firefighters in realistic conditions. Time to complete the task and number of wrong turns taken were the two metrics used in the tests. Firefighters that used virtual environment during their mission preparation showed measurable improvements in both metrics when compared to the control group that did not use virtual environment. The feasibility tests demonstrated that virtual environment is useful in mission preparation for shipboard firefighting, but the broader problem of general ship familiarization is also an area that can benefit from training in virtual environments. Learning one's way around a ship is a time-consuming and tedious process that can be accelerated by exploring a virtual model of the ship before reporting aboard. Our follow-on work has been to develop a multimodal virtual environment for use as a ship familiarization tool.

MULTIMODAL SHIP FAMILIARIZATION TOOL SYSTEM DESIGN

The idea of creating a 3-D virtual model of a ship to allow the user to get an idea of the ship's layout is not new, but most systems that provide this capability have cumbersome user interfaces, provide little or no feedback, or require knowledge of a specific computer-aided design (CAD) modeling program to use them. The goal of this project is to develop an easy-to-use, stand-alone, self-paced system that sailors can use as a wayfinding aid to help them rapidly develop good familiarity with the ship's spaces. This system could be used either shoreside before reporting to a ship, or as an embedded training system on the ship. The key functional components of this system are (1) the interactive virtual environment comprising a realistic 3-D model of the ship, (2) the speech-based natural language command interpreter that provides a natural, intuitive way for users to control and query the virtual environment, and (3) the automated route planning algorithm that provides general wayfinding and path display capabilities.

The system comprises two major hardware components. The VR application runs on an Intergraph 500 MHz Pentium III computer and provides the display and manual user interface that is used for navigation in the virtual environment. The speech analysis and interpretation and the route planning are performed on a Dell 366 MHz Inspiron 7000 laptop computer. This arrangement provides a degree of flexibility for program development, allowing the two major components to be developed independently. Having a separate display for speech processing feedback is also very helpful, especially during the debug process. Interprocess communication between the two components uses TCP/IP sockets. Figure 1 illustrates the modular system design.

The software was developed using commercial-off-the-shelf (COTS) software development kits (SDKs). The VR application uses the World Toolkit from Engineering Animation, Inc. This is a cross-platform software development system for building 3-D applications. This product was selected because our intent was to develop software to run on both a Silicon Graphics O2 and an Intergraph Windows NT OpenGL computer, and this SDK supported both systems. Software drivers are provided for a variety of user interface devices, but our experience showed that support for these devices was not always complete, and the SDK did not provide all the capabilities we needed. Our development started with the O2 and was converted to the Intergraph, and this conversion was not difficult because of World Toolkit's cross-platform support.

The speech recognition system used for this project was IBM's ViaVoice Executive. This is a speaker-independent continuous speech recognition system that requires minimal training and supports dictation, command-line, and grammar-based vocabularies. The ViaVoice SDK provides the tools for specifying utterance labels or parse tags to be output by the recognizer, thereby simplifying the implementation of the command interpreter and allowing us to develop a more sophisticated interface without the need for a complete natural language understanding system.

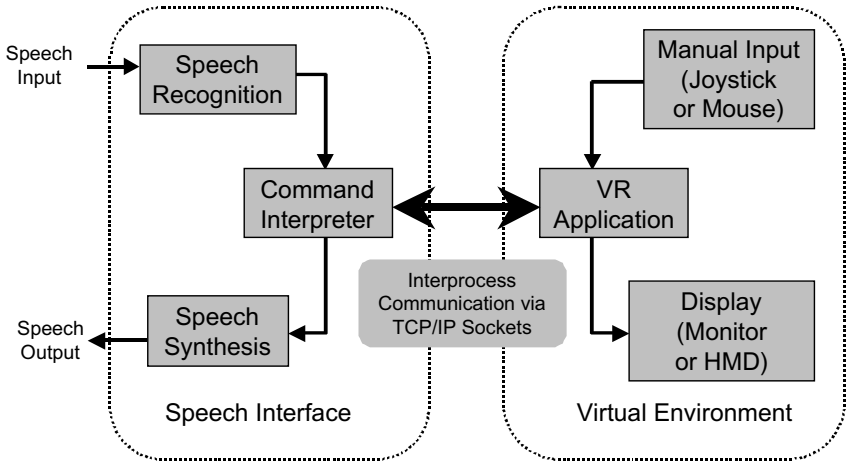


Fig. 1 — The modular system design provides flexibility for software development

Virtual Environment of the Ex-USS *Shadwell*

Earlier virtual models of the *Shadwell* were developed to meet specific requirements for a particular firefighting exercise or proof-of-concept demonstration. The first portion of the ship that was modeled was a single main machinery space that comprised only two compartments and required less than 2,500 polygons. This model was used primarily for a VR walk-through proof-of-concept and included a fractal model of a fire [1]. The second VR project involving the *Shadwell* was a feasibility study to evaluate the effectiveness of using immersive virtual environments as an aid to shipboard firefighters described earlier in this report and in Refs. 2, 5, and 6. For that project, only the portions of the ship that were used during the test were modeled. These portions comprised approximately 25 compartments with 28 operable doors, creating a model of approximately 4,800 polygons.

For our familiarization project, it was necessary to create a model of the *Shadwell* that included all the areas of the ship to which the crew had access. This includes staterooms, berthing spaces, mess areas, storerooms, electronics spaces, machinery rooms, weather decks, heads, and passageways. For the *Shadwell*, there are 147 compartments that meet our requirements. A highly detailed model of a structure of this size could potentially create millions of polygons that would require a high-end graphics computer to display effectively. Since a potential use for this system would be for a self-paced training system running on a desktop computer, we created a model that was relatively simple in terms of geometric complexity, yet provided a realistic representation of the ship. Figure 2 shows how this technique can provide a realistic view from the quarterdeck of the virtual model.

To reduce the load on the graphics rendering engine and thereby maintain an acceptable frame rate on the display, some effort was taken to create simple, yet realistic models of various components. Model complexity was reduced by omitting fine details of the ship's construction, such as support beams and stiffeners, and by simplifying the representation of the roughly 400 doors in the *Shadwell*. An example of this is the modeling of the watertight doors. The actual doors are oval shaped with a wheel in the center that operates the complicated latching mechanisms. A detailed model of a single door may have hundreds of polygons if modeled in fine detail, but there was no need to represent the opening mechanisms at all, and the corners of the doors were chamfered instead of rounded. Texture map images that showed the wheel and latching mechanism were applied to the faces of the doors to give the correct appearance, while significantly reducing the geometric complexity. This created a realistic looking door that was not a burden to the graphics processor when replicated hundreds of times. Our efforts were successful in creating a model having 65,000 polygons that could be rendered at a minimum of six frames per second on an Intergraph 500 MHz

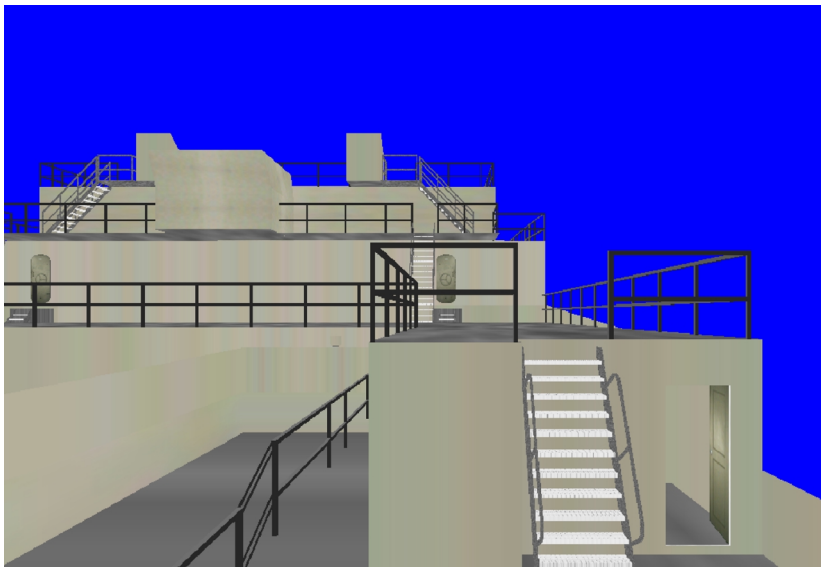


Fig. 2 — A view from the quarterdeck of the virtual *Shadwell* model

Pentium III computer. The VR application also runs on a Silicon Graphics O2 computer at slightly slower frame rates.

The *Shadwell* was constructed before CAD tools were available to naval architects, so no model was readily available that we could import into our VR system. Because of the ship's on-going role as a research vessel, however, accurate CAD files had been developed for many of the test areas of the ship. For our project, these files (some of which were only 2-D plan views) were expanded and combined to create a complete 3-D model. As with many VR projects, the conversion from a CAD format to a format that could be read with our VR application was not straightforward. Typically, some detail can be lost when converting from one format to another, and the texture maps that appear correct in one program are displayed incorrectly when converted to another. During our project, all of the tools we used for 3-D modeling, file conversion, and the VR application were updated by the manufacturers. This created a version compatibility problem that was unavoidable because many of the updates provided capabilities that we needed or corrected problems that we encountered.

The *Shadwell* model comprises several modules that contain various components of the ship. The modules are generally divided into individual decks, but some contain other items, such as ladders or handrails. A configuration file is used to specify which modules are loaded during program initialization. This allows the user to change the virtual environment without recompiling the application. With this mechanism, the user could actually use the VR application to explore a completely different model. The file also allows up to ten locations in the model to be used as predefined viewpoints that are accessible with keyboard commands. This is useful for setting viewpoints at locations of frequent use or of interest for specific demonstrations. To allow manipulation of individual doors, they must be specified in the configuration file. The location, type, name, and direction and amount of swing are specified for either watertight or ordinary (nontight) doors.

The user interface for the virtual environment supports both the desktop monitor and the head-mounted display (HMD). When using the desktop monitor, the mouse and keyboard control interaction with the environment. The left mouse button provides forward or backward movement in any direction at variable speeds based on the cursor (i.e., mouse) position, and the right mouse button provides lateral movement. Simultaneously pressing the left and right buttons changes the pitch of the viewpoint so the user can look up

or down. The middle mouse button provides a mechanism for emulating a push-to-talk function to the speech interface. This eliminates the requirement to use the mouse on the speech interface PC to signal the beginning and end of an utterance, thereby letting the user control both the VR application and the speech interface with a single mouse.

Keyboard commands provide users with additional interactions or allow them to change configuration parameters. The numeric *0-9* keys move the user position to the ten predefined viewpoints described earlier. To operate individual doors, the user positions the cursor over the door and presses the *Tab* key. All doors can be opened simultaneously with the *O* key, or closed with the *C* key. Terrain following is enabled and disabled with the *T* key. Terrain following causes the user viewpoint to remain a fixed distance above the decks and ladders giving the appearance of walking around. When terrain following is disabled, vertical lateral movement is possible, giving the appearance of flying. The + and – keys increase and decrease the apparent viewpoint height. The *D* key enables and disables collision detection. Collision detection prevents the user from walking through walls, which means they must open doors and move through the model as they would have to move through the real ship. Collision detection can be disabled to allow users to escape from areas they get into and can't manage to get out of (such as under inclined ladders). Terrain following and collision detection are both enabled during program initialization. The user's viewpoint pitch angle can be changed with the mouse as described earlier, and resetting it to the normal straight-ahead viewing angle is accomplished with the = key.

When using the HMD, a different mechanism controls interaction with the environment. A hand-held joystick has buttons to control movement and to operate doors. Pressing the *Forward* or *Backward* buttons controls movement in the respective directions. Movement occurs in the direction the user is looking. The *Activate* button operates any door at which the user is looking. The keyboard commands are still available, but they are typically not used when wearing the HMD.

Error messages are displayed in a separate window when the user tries to perform an invalid action. All messages include the ASCII *Bell* character, which causes the computer to make a beep sound. Typical messages are *That object doesn't do anything* when trying to activate an object that isn't a door, or simply *Bonk*! when a collision is detected. When using the HMD, users cannot see the message, but after hearing the beep sound, they can usually determine what the invalid action is.

Spoken Natural Language Interface

One of the biggest drawbacks to the system we used for our feasibility tests was its rudimentary human-computer interface. A magnetic tracking system followed the user's viewpoint and hand movement, and joystick buttons provided navigation capability, but there was no way for the user to obtain any information about the virtual environment. Thus the model was simply a passive space to be explored, where the only information available was the numbers shown on the doors—there was no way to find out anything about the rooms or to get help if you were lost.

To overcome this drawback, a speech-based interface was developed for the familiarization tool. This interface enables the user to interact with the virtual environment in a natural and intuitive way even while wearing an HMD. This improves the usefulness of the system by allowing the user to ask questions of the virtual environment system and by providing informative feedback via synthesized speech; it also gives the user a powerful means of accessing information about the virtual world. Rather than relying on ad hoc exploration, as is required in most video games, the user of the interactive familiarization tool is able to ask, for example, "Where is Repair 2?" "Which way is forward?" or "Which deck am I on now?" Commands can also be issued, such as, "Put me on the forecastle" or "Show me how to get to the Control Room." Such

intuitive interaction provides much more flexibility than conventional 3-D programs, and the immersive environment allows users to get a much better feel of the real layout of the ship than they can get from 2-D diagrams.

A context-free grammar developed specifically for this application defines the set of utterances to be recognized by the interface; it uses a vocabulary of approximately 170 words and recognizes over 16,000 utterances. Using this interface, a person can request information about the name, number, and location of the room he/she is currently in ("What's the number of this compartment?") or about any of the other rooms in the database ("Which deck is the Communications Center on?"). The user can also request information about his/her current location ("Where am I?" "Which way am I facing?") or ask to be transported directly to any room ("Put me in the Crew Mess.").

Commands and queries from the user are identified by the speech recognizer and converted to one of 20 parse tags according to specifications written into the grammar. Utterances are tagged by function, based on what the user has said, and key elements such as the room name are identified. In Table 1, the example text shows the utterances that correspond to each of the tags. The notation *(a | b | c)* indicates that either *a* or *b* or *c* is required; square brackets indicate optional words or phrases, with *[a | b]* indicating that either *a* or *b* is allowed but neither is required. For example, *(Take | Fly) me [from here to] there* indicates that "Take me there," "Take me from here to there," "Fly me there," and "Fly me from here to there" are all allowable utterances. Tags that have parameters are indicated by -> *Tag {Parameter}* where the parameter is extracted from the text of the input utterance. *{ROOMNAME}* is the name of one of the 147 compartments included in the model, and *{DIRECTION}* is *forward*, *aft*, *port,* or *starboard*.

The tagged utterances are passed to the command interpreter, which processes the tags, retrieves the necessary information, and generates an appropriate response. Each time an utterance is processed, the room that is the current topic of conversation is stored to allow the user to ask follow-up questions about the same room using anaphoric reference ("it," "there") rather than having to specify the room's name or number each time. To determine which compartment the user is in, the command interpreter requests the coordinates of the user's current location from the VR application; the *z* coordinate, or altitude, indicates which deck he or she is on, and a ray-crossing algorithm is used to calculate which compartment (as defined by the set of vertices comprising its perimeter) contains the *x,y* point.

The verbal responses generated by the command interpreter are matched to the syntactic structure of the user's question. For example, if the user asks, "Which room am I in?" the system responds, "You are in the X." If the user asks, "Which room is this?" the system responds, "This is the X." Using complete sentences as feedback accomplishes two things: it encourages the user to also speak in complete sentences as required by the recognition grammar and it helps the user realize when the recognizer has made an error.

One of the primary challenges in developing this interface was the fact that the ship model did not contain any information about the compartments; it was merely a collection of graphical elements. In order to reference specific rooms using natural language, it was necessary to construct a supplemental database containing the name and number of each room, along with the coordinates that comprise the room's perimeter, so that it would be possible to determine in which room the user was located. This was necessary to convert the compartment names and/or numbers used by the speech interface into (*x,y,z*) coordinates used by the VR application.

The command interpreter in this interface performs no linguistic analysis of the utterance; it simply identifies word and phrase patterns. This approach is suitable for this particular application because of the limited vocabulary size and simple linguistic structure of the utterances, but it would probably not be

Table 1 — Utterance Tags and Example Text

Tag	Example text
ThisName	(what's | what is) the name of this (room | compartment) (what | which) (room | compartment) is this
ThisNameIn	[now] where am I [now] (what | which) (room | compartment) am I in [now] what is my [current] location
ThisNumber	(what's | what is) the [room | compartment] number (of | for) this (room | compartment) (what's | what is) the [room | compartment] number here
ThisDeck	(what | which) deck is this
ThisDeckOn	(what | which) deck am I on
TheName	(what's | what is) (its | the) name (tell | give) me (its | the) name
TheNumber	where is it (what's | what is) (its | the) [room | compartment] number (tell | give) me (its | the) [room | compartment] number
TheDeck	(what | which) deck is it on (what's | what is) (its | the) deck (tell | give) me (its | the) deck
OtherNumber	(what's | what is) the [room | compartment] number (of | for) the {ROOMNAME} -> OtherNumber {ROOMNAME} (where's | where is) the {ROOMNAME} -> OtherNumber {ROOMNAME}
OtherDeck	(what | which) deck is the {ROOMNAME} on -> OtherDeck {ROOMNAME}
WhichWay	which (way | direction | side) is {DIRECTION} -> WhichWay {DIRECTION} (where's | where is) {DIRECTION} -> WhichWay {DIRECTION}
Facing	[now] which way am I facing [now]
Transport	put me in the {ROOMNAME} -> Transport {ROOMNAME} put me (in it | there) -> Transport there
FollowPath	(take | fly) me [from here] to the {ROOMNAME} -> FollowPath here, {ROOMNAME} (take | fly) me to it -> FollowPath here,there (take | fly) me [from here to] there -> FollowPath here,there (take | fly) me from the {ROOMNAME1} to the {ROOMNAME2} -> FollowPath {ROOMNAME1},{ROOMNAME2} follow the (path | arrows) -> FollowPath here,there
StopFollow	stop (flying | following [the path])
ShowPath	(how do I | (show | tell) me how to) get [from here] to the {ROOMNAME} -> ShowPath here, {ROOMNAME} (show | tell) me how to get [from here to] there -> ShowPath here,there how do I get there -> ShowPath here,there (how do I | (show | tell) me how to) get there from here -> ShowPath here,there (how do I | (show | tell) me how to) get there from the {ROOMNAME} -> ShowPath {ROOMNAME},there (how do I | (show | tell) me how to) get from the {ROOMNAME1} to the {ROOMNAME2} -> ShowPath {ROOMNAME1},{ROOMNAME2}
HidePath	(hide | turn off | stop showing) the (path | arrows) turn the (path | arrows) off
ArrowsOn	(show | turn on) the (path | arrows) turn the arrows [back] on
Help	hello help [please] [[can you] give me] instructions [please] what (can | do) I (say | do) [now]
Quit	(quit | exit | stop | kill | end) the (program | simulation) shut down the (program | simulation)

sufficient to support more complex interactions. Further information about this interface can be found in Refs. 8, 9, and 10.

Automated Route Planning and Wayfinding

In addition to asking questions about the compartments and his/her location, the user can tell the system to indicate the route to a particular compartment ("Show me how to get to the Control Room.") or between any two compartments ("How do I get from the Control Room to the Communications Center?"). Compartments and doorways are represented as nodes in a network in which the adjacency and connectivity information is stored. Additional path correction nodes are added where necessary when straight-line paths between nodes are obstructed by bulkheads or where those paths would cause the user to cross open areas (such as the well deck). Our system requires 326 nodes to accommodate the paths between the 147 compartments. The automatic wayfinding algorithm, or route planner, uses the adjacency network to determine the best path from one node to another.

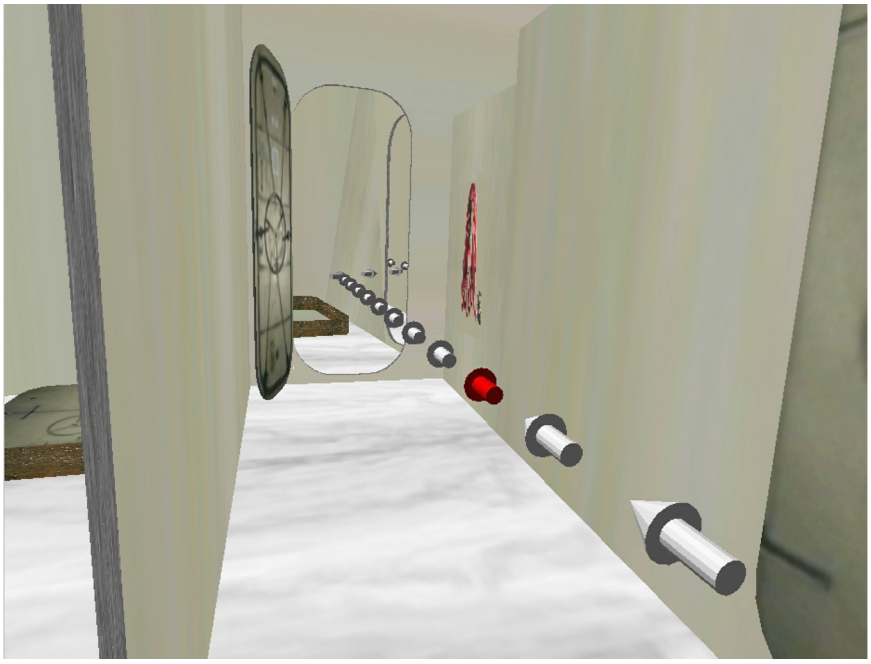
The route-planning algorithm used in this system is based on a LISP implementation by Michael L. Hilton used in the Land Air Combat in Eric (LACE) military simulation system [11] to generate on-road routes for ground vehicles. The search routine does a breadth-first exploration of the adjacency network starting from the point of origin and then generating incrementally longer subpaths until a path to the destination is found. Route quality is improved by providing a user-defined cost function that ranks intermediate results and expands the most promising subpaths first. The cost function used in this project simply minimizes the total cost of all the nodes comprising the route. Assuming that the nodes are more or less evenly spaced throughout the ship, this cost function would always find the shortest path between two points if all nodes had equal cost. However, by making some nodes more expensive than others, we can fine-tune the search to yield slightly longer paths that meet additional quality criteria, such as avoiding certain compartments or going out on deck only when necessary. For example, the nodes corresponding to doors that connect interior and exterior spaces were assigned a relatively high cost so that routes originating indoors tend to stay indoors, and those originating on the deck tend to remain outdoors whenever practical.

Once the complete path has been determined, the list of waypoints corresponding to the locations of the nodes along the path is sent to the VR application. It then displays the path as an animated sequence of arrows pointing toward the destination as shown in Fig. 3. The arrows can be selectively hidden or shown to allow the user to use the visual aid only when he/she requests it. All compartments in the database can be the beginning or end of any path, and special segments are included in the path to avoid obstacles where necessary. By commanding the system to "Take me there," the user can "ride" along the path under computer control.

Interprocess Communication

The VR application and the speech interface run on different computers, so an interprocess communication protocol was developed using TCP/IP stream sockets to allow data transfers between the two systems. The VR application provides a server function while the speech interface acts as a client. The user starts the VR application first, and then starts the speech interface program. The VR application runs as described above while waiting for a connection request from the speech interface.

Once the TCP/IP connection is established, the VR application and the speech interface communicate by using a simple text-based protocol. The primary request sent from the speech interface to the VR application is *getlocation*. This query is issued whenever the speech interface needs to make a determination of the user's location and/or viewing direction. The VR application responds with *location x y z h p r*. The first



Click on Fig. 3 to view Quick Time movie.

Fig. 3 — Arrows show the path computed by the route planner

three parameters are the 3-D (*x,y,z*) position of the user in the model, and the *h*, *p*, and *r* parameters provide the heading, pitch, and roll of the viewpoint. This query and response are exchanged quite often, even when it is not apparent from the conversation that it is required. This is because the user is free to wander about in the model during conversations with the speech interface, and there is an implied "here" in many sentences (such as, "How do I get [from here] to the Quarterdeck?").

Moving the user to a new location in the model is accomplished by sending *setlocation x y z h p r* to the VR application. The *setlocation* command has the same parameters as the *getlocation* query. The effect on the VR application is to instantaneously change the location and viewing direction to the new settings.

Paths can be defined by sending *waypoint* and *addwaypoint* commands to the VR application. When the VR application receives a *waypoint* command, it deletes all previous waypoints (if any have been set) and constructs a new list of waypoints. The *addwaypoint* command appends waypoints to an existing waypoint list. The format for both commands is *[add]waypoint x1 y1 z1 x2 y2 z2 x3 y3 z3*, where (*x1*,*y1*,*z1*) is the location of the first waypoint, (*x2*,*y2*,*z2*) is the location of the second waypoint, etc. Any number of locations can be included in the commands, and typically even seemingly short paths may have numerous waypoints. Waypoints are assumed to be locations on the deck that define the path. The *showpath/hidepath* commands cause the VR application to display/hide the animated arrows shown in Fig. 3 at a height of 24 inches above the deck. The *followpath* command causes the VR application to move the user location along the path at approximately walking speed. Interpolation between waypoints is performed to cause the user movement to have a constant, smooth movement. The *stop* command is issued to terminate the automated path following process.

The interprocess communication protocol also supports operation of the doors. The *openalldoors* and *closealldoors* commands operate all doors simultaneously. Individual doors can be operated using the *opendoor door_name* and *closedoor door_name* commands. The *door_name* parameter is the name assigned to the door when it is loaded via the configuration file as described above. When following a path, *opendoor*

commands are sent to open all the doors along the path, to permit realistic movement through the ship, and to prevent the unnerving effect of crashing through a closed door.

The speech interface accepts the *say* command to provide a mechanism for giving verbal feedback to the user. The VR application currently uses this command to indicate that the path following process has completed by issuing *say You have reached your destination*, although a variety of other uses is expected in the future.

To view a QuickTime® movie of the demonstration of this interface, click here.

FUTURE CONSIDERATIONS

Object and Information Representation

One of the problems that we confronted in this project was the lack of a unified method of object and information representation for the virtual world. To date, most virtual worlds have been designed primarily for looking at—users move in and around objects and scenery that convey information almost exclusively through the visual channel. In some applications, users hear sounds to supplement the visual information, and sometimes the user can click on an object to trigger some sort of action (typically an animation sequence), but still the emphasis is on the visual.

When we add speech recognition and language processing capabilities to the VR interface, we move beyond the limited visual interaction and enable the user to access a much wider range of information. In order to support this enriched interaction, the natural language processor needs access to information that is often abstract or is not important to the visual presentation of the scene, such as the name of the objects to which the user can refer, information about part-whole relationships, and relative location information (i.e. what is next to what). Typically, none of this information is contained in the database for a VR system, and it may be difficult to obtain from the visual representation alone.

Object representation was one of the primary challenges in developing the interface for this system. Because the VR model contained no representation of the compartments as objects, it was necessary to construct a supplemental database containing the name and number of each compartment in order to reference the individual rooms. The database also needed to contain the coordinates of all of the vertices that comprise each room's perimeter to enable us to determine in which room a specific point is located. While this solution worked well for this system, it is unlikely that it would be viable for large or complex projects.

For VR and natural language to be integrated and work well together, the VR building tools must include some way of enriching the VR world with the information needed by the language processing module. Ideally there would be a shared data structure that includes all of the objects in the virtual world, with information about each object's location and visual representation, as well as information about what it is, what its name is (if appropriate), etc. This database would keep track of which objects are visible at any given time and could be queried for answers to the user's questions about the various things he or she sees in the virtual world. The VR model used in this system actually has the beginnings of such a data structure—it contains a list of all the doors in the model, implemented as nodes in the graphical hierarchy. This allows users to click on the doors to open or close them and allows the command interpreter to open or close individual doors by name.

This type of integrated, shared data structure would also help minimize the number of different and

possibly overlapping sets of information required in a complex VR system. This integration will become increasingly critical as VR applications become more interactive and users are able to alter the VR world by moving objects to different locations, creating things, or destroying them. If a dynamic, interactive VR system does not use an integrated database, it will be difficult to ensure that all information is properly updated as changes occur, which will have an adverse effect on the usability of the system.

Dynamic Path Costs

For this project, path segments were assigned fixed cost values that assume that paths between nodes remain static. Although the route planner takes into account the cost of each segment, there is no mechanism for the user to change the relative cost of any segment. This capability needs to be added so that users can direct the route planner to avoid (or include) certain areas of the ship. This would be particularly useful in battle damage situations where part of the ship is damaged and the path that one would normally use is blocked. Cost assignments could be made semipermanent by saying "I can't go through CSMC" or "Door 1-13-0 is blocked"; or they can be performed on a per-request basis as in "Show me how to get to the Data Analysis Room without going through DC Central." This capability would greatly enhance the flexibility of this system and would make it more in line with its expected use.

Greater Availability to Users

For this project, we used COTS SDKs and high-end PCs to get the performance we needed for our prototype system. For universal accessibility, a familiarization tool that can be used on any computer would be necessary. Providing this capability to anyone with access to the Internet would be ideal for providing maximum dissemination. Work is already under way to enable Web browsers to provide interaction with and display of 3-D objects on the user's screen. Tools such as the Virtual Reality Modeling Language (VRML) and the Java Virtual Machine are beginning to make small, simple, interactive virtual environments available to anyone with a Web browser. Our current requirements for system performance can't be met by Web browsers, but these new Internet tools may provide a mechanism for less capable systems to get some of the functionality of our system. Research needs to be done to determine how to use these tools without incurring severe performance penalties, so that our familiarization tool could be readily available to anyone who needs it.

Utilization of Shipbuilder Data

All shipbuilders today use CAD programs and databases of one form or another to design and build their new ships. One of the problems we encountered was that no computerized version of ships' drawings was available for building our virtual model because of the age of the vessel. This would not be true for ships that are currently in the design phase, but unfortunately there is no standardized format that all ship manufacturers use. Although file conversion utilities are available for many products, we have found that this conversion process often results in some loss of data. It would be useful to have a standard format for virtual models that preserves all the detail included in the original design. A method of gleaning information from databases that can be used by the speech processing system would also be beneficial.

Many CAD programs provide some type of 3-D visualization to display the drawings as a virtual model. Although this may provide an effective display when a naval architect who uses the CAD program every day is manipulating the image, the inexperienced user often finds this technique to be difficult for viewing any particular portion of the ship. Our system provides an intuitive way of exploring the ship in ways the user

finds familiar. Importing the virtual model created by the shipbuilders into our multimodal system would provide the kind of functionality and flexibility that would make an excellent ship familiarization tool.

CONCLUSION

We have successfully developed and demonstrated a multimodal immersive virtual environment for ship familiarization by using a combination of VR, speech recognition, natural language processing, route planning, and speech synthesis technologies. This system can be used as an automated stand-alone, self-paced system that sailors can use as a wayfinding aid to help them rapidly develop good familiarity with the ship's spaces. Although this system is suitable for the scope of this project, we have indicated areas that require follow-on efforts to address the more general problem of making this technology available to all ship types and classes. Further work is required in these areas to create a familiarization tool that could be used to its fullest potential.

ACKNOWLEDGMENTS

The authors thank Mr. Charles Bogner of the Office of the Chief of Naval Operations, Surface Warfare Division, Safety and Survivability Branch (N86DC), for his support of this effort. Special thanks goes to Scott Fowler and Brad Havlovick of Havlovick Engineering Services for their construction of the virtual *Shadwell* model, and to Dr. Fred Lepple of N86DC for contract administration.

POINTS OF CONTACT

David Tate, david.tate@nrl.navy.mil, Code 5523, Naval Research Laboratory, Washington DC 20375
Stephanie Everett, everett@aic.nrl.navy.mil, Code 5512, Naval Research Laboratory, Washington DC 20375
Tucker Maney, maney@aic.nrl.navy.mil, Code 5512, Naval Research Laboratory, Washington DC 20375
Ken Wauchope, wauchope@aic.nrl.navy.mil, Code 5512, Naval Research Laboratory, Washington DC 20375

REFERENCES

1. L. Rosenblum, J. Durbin, U. Obeysakare, L. Sibert, D. Tate, J. Templeman, J. Agrawal, D. Fasulo, T. Meyer, G. Newton, A. Shalev, and T. King, "Shipboard VR: From Damage Control to Design," *IEEE Computer Graphics and Applications* **16**(6), 10-13 (1996).

2. D. Tate, L. Sibert, and T. King, "Using Virtual Environments for Firefighter Training," *IEEE Computer Graphics and Applications* **17**(6), 23-29 (1997).

3. B. Witmer, J. Bailey, and B. Knerr, "Virtual Spaces and Real World Places: Transfer of Route Knowledge," *International Journal of Human-Computer Studies* **45**, 413-428 (1996).

4. J. Bliss, P. Tidwell, and M. Guest, "The Effectiveness of Virtual Reality for Administering Spatial Navigation Training to Firefighters," *Presence* **6**(1), 73-86 (1997).

5. D. L. Tate, L. Sibert, and T. King, "Virtual Environments for Shipboard Firefighting Training," Proceedings of the Virtual Reality Annual International Symposium (VRAIS '97), March 1997, Albuquerque, New Mexico, pp. 61-68.

6. F. Williams, P. Tatem, J. Farley, D. Tate, L. Sibert, T. King, D. Hewitt, C. Siegman, J. Wong, and T. Toomey, "Virtual Environment Firefighting / Ship Familiarization Feasibility Tests," NRL/FR/6180--

97-9861, Naval Research Laboratory, September 30, 1997. See also http://www.chemistry.nrl.navy.mil/damagecontrol/vr.html

7. H.W. Carhart, F.W. Williams, and T.A. Toomey, "The Ex-Shadwell – Full Scale Fire Research and Test Ship," NRL Memorandum Report 6074, Naval Research Laboratory, October 1987 (reissued September 1992).

8. S. S. Everett, K. Wauchope, and M. A. Pérez Quiñones, "Creating Natural Language Interfaces to VR Systems: Experiences, Observations and Lessons Learned," Proceedings of the 4th International Conference on Virtual Systems and Multimedia (VSMM98), November 1998, Gifu, Japan,Vol. 2, pp. 469-474. See also http://www.aic.nrl.navy.mil/~severett/VSMM98/VSMM98.html

9. S. S. Everett, K. Wauchope, and M. A. Pérez Quiñones, "Creating Natural Language Interfaces to VR Systems," *Virtual Reality*, **4**, 103-113 (1999).

10. K. Wauchope, S. S. Everett, D. Perzanowski, and E. Marsh, "Natural Language in Four Spatial Interfaces," Proceedings of the Fifth Conference on Applied Natural Language Processing, 1997, Washington, D.C., pp. 8-11. (A postscript version of this paper is available at ftp://ftp.aic.nrl.navy.mil/pub/papers/1997/AIC-97-020.ps

11. Craig S. Anken, "LACE: Land Air Combat in Eric," RADC-TR-89-219, Rome Air Development Center, October 1989.